Lead-Lag Analysis via Sparse Co-Projection in Correlated Text Streams

Fangzhao Wu[†], Yangqiu Song[§], Shixia Liu[‡], Yongfeng Huang[†], Zhenyu Liu[♮]
[†]Tsinghua National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing, China
[§]Hong Kong University of Science and Technology, Hong Kong
[‡]Microsoft Research Asia, Beijing, China
[♮]China International Communication Center, SCIO, Beijing, China
wufangzhao@gmail.com, yqsong@gmail.com, shixia.liu@microsoft.com, yfhuang@tsinghua.edu.cn, liuzhenyu@cicc.org.cn

ABSTRACT

Correlated topical trend detection is very useful in analyzing public and social media influence. In this paper, we propose an algorithm that can both detect the correlation and discover the corresponding keywords that trigger the correlation. To detect the correlation, we use a projection vector to project two text streams onto the same space, and then use a least square cost function to regress one text stream over the other with different time lags. To extract the corresponding keywords, we impose the non-negative sparsity constraints over the projection parameters. In addition, we present an accelerated algorithm based on Nesterov's method to efficiently solve the optimization problem. In our experiments, we use both syntehtic and real data sets to demonstrate the advantages and capabilities of the proposed algorithm over CCA on the follower link prediction problem.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*

General Terms

Algorithm, Experimentation

Keywords

Lead-Lag Analysis; Correlation Detection; Gradient Descent

1. INTRODUCTION

With the explosive growth of online information such as news articles, blog posts, microblog posts and so on, extensive studies have been conducted to help users better understand and consume this textual information. One interesting problem is detecting whether the content from different resources is correlated and impacts each other. For example, some news providers publish news on certain topics faster than others [9]. Another example is topic influence in a social network. Commonly, the topics of interest of a user in a social network tend to be influenced by their friends' topics [20]. Thus a method to characterize and compare topic trending behaviors based on the text content is very useful for analysts to identify lead topics as well as their dissemination and impact over time.

To help users examine topical lead-lag relationships across corpora, Shi *et al.* [17] proposed an intuitive and simple solution based on LDA [3] and time series analysis. Although this method has achieved a certain amount of success in comparing topic trending behaviors between two text corpora, there are still two challenges to be addressed.

One challenge is to effectively detect and model the correlation between text streams. The text content is usually represented by bag-of-words. Thus, the data is very high-dimensional. Even if we pre-process the text with a topic model or a clustering method, e.g., latent Dirichlet allocation (LDA) [3] or a constrained coclustering approach [18], the space representing the topics is still high-dimensional in the statistical sense. For example, consider that we have 100 topics for two text streams. If we want to analyze their daily correlation for one year, we only have 365 time stamps to do the correlation analysis. Compared to a 100 dimensional space, the 365-timestamped data is still too small.

Another challenge is to find keywords that trigger the correlation. For example, we have two text streams discussing the US presidential election. Assume we analyze the correlation on the original keyword space instead of the topic space. Typically, keywords such as "Obama", "Romney", "economy" and "war" may trigger the correlation. Other keywords such as "Indonesian", "Harvard", and "businessman" may undermine the correlation score. It is not trivial to automatically detect the causal keywords, since there may be a lot of noise.

Recently, a canonical correlation analysis (CCA) based method was proposed to detect the correlation of trends [2]. This method works well in detecting the canonical trend between text streams. However, it still faces some problems. First, CCA uses two projection vectors for each of the two text streams, and analyzes the correlation in the projected one-dimensional time series. As a result, it can detect various kinds of correlations, even when the topics of two text

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

http://dx.doi.org/10.1145/2505515.2505554.

streams are not relevant. In practice, when analyzing the topical lead-lag correlation for the influence on some specific topics, we do not want to include the correlation across topics that are totally irrelevant. Second, CCA will detect both positive and negative correlations. When some of the keywords are positively correlated and others are negatively correlated, CCA cannot distinguish them. However, we do not expect that the lead-lag is caused by a mixture of both positive and negative correlations.

In this paper, we propose an algorithm to both analyze the lead-lag correlation between text streams and extract the keywords that lead to such correlation. To analyze the correlation, we only use one projection vector to project the two candidate text streams onto two time series, and use a least square cost function between two time series to evaluate the correlation. To extract the leading keywords, we constrain the projection vector to be non-negative, and set the L_1-norm to one, i.e., making the projection vector lie in the probabilistic simplex. As a consequence, only keywords that are positively correlated have non-zero values of the elements in the projection vector. Thus, the projection vector also performs as a feature selection to find the leading keywords. We solve the optimization problem using an accelerated gradient algorithm based on Nesterov's method [13], which is fast and suitable for large-scale data analysis.

We conducted experiments on two sets of synthetic data to show the difference between our algorithm and CCA, and also use another synthetic dataset with different scales to demonstrate the correctness and efficiency of the accelerated algorithm. Moreover, we present three interesting case studies on social media data based on the application of our algorithm:

- **Common trend detection and keyword selection:** In social media, it is interesting to analyze whether two users have common preferences and interests, especially over time. The most intuitive way is to find common topics or keywords that trigger the correlation between their message text streams. Our method can effectively extract the keywords and find the lead-lag correlation. Given two correlated users, if one user does not publish any messages but the other has a lot, we can automatically recommend some messages using the selected keywords to best meet the user's interests.
- **Trend setter detection:** In social media, it is useful to automatically find the information leaders. Information leaders can influence their followers' opinions, interests and even their ethical orientation. But there are too many such users in a social network and it is hard to manually label them. One obvious pattern of information leaders is that they spread information before average users do, and their messages are frequently retweeted. Our approach can be directly applied to this application. We show that by combining the correlation score and the lead-lag time, we can effectively find topical trend setters on Twitter.
- **Network influence relationship analysis:** In a social network, we are not only interested in whether two users are linked, but also why there is an edge between them. Thus, we conduct a case study on social network influence relationship analysis using our method. The results demonstrate that correlation analysis is useful for link type interpretation and link prediction.

The rest of the paper is organized as follows. We introduce related work in Section 2. Then the detailed algorithm is presented in Section 3 and the connection to CCA is presented in Section 4. Next, we show the experiments that demonstrate the problems with current approaches and the advantages of our approach in Section 5. Finally we conclude our paper in Section 6.

2. RELATED WORK

In this section, we review several papers related to our method. We first briefly discuss the current development of text or topic correlation analysis, then we present some related algorithms.

Topic detection and tracking (TDT) has been investigated for years [1]. Recent research has also studied how to detect topics correlated with each other [22] and how to extract topics from different sources sharing common information over time [23, 26]. In addition to detecting correlated topics over time, it is more interesting to analyze the correlation itself. For example, given two topics, or two sets of text streams under the same topic(s), can we detect whether they are correlated and how they are correlated?

Shi *et al.* used a simply defined correlation score to analyze the lead-lag behaviors between two topic distributions [17]. However, this method can only detect the lead-lag relationships between two text streams in the context of the same topic. It also depends on the results of topic models [3]. In addition, it is unclear which topics/keywords trigger the overall correlation of two text streams. Thus, its usage in many real-world applications is limited. Steeg and Galstyan proposed an entropy based lead-lag analysis for two topic distributions, and applied it to the social network link prediction problem [19]. Their method does not consider long-term correlation between text streams. Instead, it only analyzes the local historical impact. Recently, Bießmann et al. [2] employed temporal kernel CCA to detect the canonical trend between two text streams. As we mentioned in the introduction, this method uses two projection vectors to find the correlation. It can detect correlations when two text streams have similar trends in volume, even if they are talking about totally different topics. Moreover, CCA may mix up positive correlations and negative correlations. Although these correlations may sometimes be interesting, they are probably spurious. Here we focus on detecting the positive correlation relationship between two high-dimensional text streams. We prove that the correlation found by our method is more reliable in the experiment section on both synthetic data and real-world data.

Some variants of the CCA algorithm also consider the sparsity of the projection vector to improve the interpretability of results [14, 15, 24, 8]. However, none of them imposes a non-negative constraint, and thus they all mix up the positive and negative correlations. Witten and Tibshirani [25] proposed imposing both non-negative and sparse constraints on the projection vectors. Sparse constraint has gained increasing attention and proven to have good generalization and interpretability [21]. However, this method still uses two projection vectors to find the correlation and consequently will confound different topics in the text. Furthermore, these CCA-related methods are not designed for text mining related applications. In contrast to all the above CCA-based methods, we use only one projection vector to find the correlation factors. The motivations that led us to choose

one projection vector are as follows. First, this will result in more effectiveness when the text is high-dimensional and noisy, since there are fewer parameters to estimate compared with CCA. Second, since text streams consist of homogeneous data, one projection vector will be more suitable and have a lower risk of finding irrelevant-topic correlation, and thus can reveal a more reliable relationship between the text streams.

Recently, solving a L_1-norm constraint or L_1-norm regularized sparse learning problem with the gradient method has been studied a great deal [7, 10, 4]. In [7], Ji and Ye proposed an extended gradient method and an accelerated gradient method based on Nesterov's method to solve the trace norm minimization problem. Liu *et al.* presented a similar method to handle the $l_{2,1}$-norm minimization problem [10]. In [4], Chen *et al.* applied Nesterov's method to solve a $l_{1,\infty}$ regularized sparse learning problem. When applying Nesterov's method, an important step is Euclidean projection. In [5], Duchi *et al.* proposed two efficient methods to do projections onto the l_1-ball in high dimensions. In [11], Liu and Ye reported some approaches for computing the Euclidean projections in linear time. Based on the above existing work, we also propose an accelerated gradient algorithm derived from Nesterov's method to efficiently solve our optimization problem.

3. SPARSE CO-PROJECTION

In this section, we introduce in detail our algorithm for analyzing the correlation between text streams.

We denote $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T] \in \mathbb{R}^{D\times T}$ and $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_T] \in \mathbb{R}^{D\times T}$ as two text steams, where $\mathbf{x}_t,\ \mathbf{y}_t \in \mathbb{R}^{D\times 1}$. D is the dimension of data. For text, it is the vocabulary size. T is the length of the time stamps. In this paper, $\mathbf{X}$ and $\mathbf{Y}$ are normalized to zero mean and unit variance, that is, $\sum_{j=1}^{T} x_{ij} = 0$, $\sum_{j=1}^{T} x_{ij}^2 = 1$ and $\sum_{j=1}^{T} y_{ij} = 0$, $\sum_{j=1}^{T} y_{ij}^2 = 1$, for $i = 1, 2, ..., D$.

3.1 Model

Our algorithm is inspired by the causality analysis between two time series, which is called *co-integration* [6]. The basic idea is to use one time series to regress the other. It has been claimed that some *spurious relationships* caused by correlation will be eliminated in this way [6]. If we have two time series x_t and y_t where $t = 1, \ldots, T$, the so called "Engle-Granger two-step method" models the co-integration in following form:

$$y_t = \alpha + \beta x_t + \epsilon_t. \tag{1}$$

The first step is to estimate the parameters α and β, and the second step is to do a statistic test to judge whether ϵ_t follows a random walk [6].

For high-dimensional text data, we propose projecting the two text streams onto the same one-dimensional space, using a same projection vector. Then we analyze the regression property between the two projected time series as:

$$\begin{array}{ll} \mathbf{w} = & \arg\min_{\mathbf{w}} \| \mathbf{w}^T\mathbf{Y} - \mathbf{w}^T\mathbf{X}_\tau \|_2^2, \\ s.t.: & \sum_{i=1}^{D} w_i = 1, \\ & w_i \geq 0 \ \ (i = 1, ..., D), \end{array} \tag{2}$$

where $\mathbf{w} \in \mathbb{R}^D$ and $\mathbf{X}_\tau = [\mathbf{x}_{1-\tau}, \mathbf{x}_{2-\tau}, ..., \mathbf{x}_{T-\tau}]$ is the data with a time lead τ. Since we centralize the data before regression, we do not need the constant parameter α shown in Eq. (1).

Note that we constrain every element in $\mathbf{w}$ to be non-negative and the sum of these elements to be one. There are several benefits from these constraints:

First, it is easy to verify that the objective function in Eq. (2) is convex, and the constraint region is a closed convex set. Thus, the formulation of our model is a convex optimization problem, which is relatively easy to solve. Second, since every element of $\mathbf{w}$ is non-negative, we can avoid confounding positive correlations and negative correlations of different keywords and keep only the positive correlations. Third, according to the constraint conditions, the optimal solution $\mathbf{w}^*$ lies in a probabilistic simplex. Thus, every element in $\mathbf{w}^*$ can be interpreted as a weighting factor or probability of the corresponding keyword. The optimal solution $\mathbf{w}^*$ is a probability distribution of different keywords and can be interpreted as a common topic between these two text streams. Fourth, the constraint in our model is a special instance of L_1-norm constraint. According to Lasso [21], the optimal solution for our model is sparse and interpretable, which is useful for extracting the keywords that contribute most to the correlation between two text streams.

3.2 An Accelerated Algorithm

In this subsection we present an accelerated algorithm based on Nesterov's method [13] to solve our optimization problem in Eq. (2). For simplicity's sake, we denote $\mathbf{A}_\tau = (\mathbf{Y} - \mathbf{X}_\tau)(\mathbf{Y} - \mathbf{X}_\tau)^T \in \mathbb{R}^{D\times D}$, and our model in Eq. (2) can be rewritten as:

$$\begin{array}{ll} \arg\min_{\mathbf{w}} & f = \mathbf{w}^T\mathbf{A}_\tau\mathbf{w}, \\ s.t.: & \sum_{i=1}^{D} w_i = 1, \\ & w_i \geq 0 \ \ (i = 1, ..., D). \end{array} \tag{3}$$

Note that the objective function in our model is a differentiable convex function and the constraint region is a closed convex set. Thus, the optimization problem is a constrained smooth convex optimization problem, which can be solved by gradient descent method or subgradient descent method. However, the convergence rate of the gradient descent method and subgradient descent method are $O(1/k)$ and $O(1/\sqrt{k})$ respectively [10], where k is the iteration steps. Both $O(1/k)$ and $O(1/\sqrt{k})$ are unsatisfactory for large-scale applications. Recently, a special kind of gradient descent method called Nesterov's method has gained increasing attention and proven to be effective in solving L_1-norm regularized optimization problems [10, 4, 11]. Nesterov's method has a convergence rate of $O(1/k^2)$, which is much faster than the gradient descent and subgradient descent methods. Therefore, we introduce Nesterov's method into our model and propose an accelerated algorithm.

We denote a general constrained smooth convex optimization problem as:

$$\arg\min_{\mathbf{x}\in\mathbf{Z}} \quad f(\mathbf{x}), \tag{4}$$

where $f(\mathbf{x})$ is a differentiable convex function in $\mathbf{Z}$ and $\mathbf{Z}$ is a closed convex set.

In contrast to the gradient descent and subgradient descent methods, which only utilize the latest point to estimate the current point, Nesterov's method utilizes the last two points. It iteratively updates two points, $\mathbf{x}_i$ and $\mathbf{s}_i$, which are the approximate point and search point at the i_{th} iteration respectively [10].

The search point $\mathbf{s}_i$ is a linear combination of $\mathbf{x}_i$ and $\mathbf{x}_{i-1}$:

$$\mathbf{s}_i = \mathbf{x}_i + \alpha_i(\mathbf{x}_i - \mathbf{x}_{i-1}), \tag{5}$$

where α_i is a combination coefficient at the i_{th} iteration. The approximate point $\mathbf{x}_{i+1}$ is updated as:

$$\mathbf{x}_{i+1} = Ep_{\mathbf{Z}}(\mathbf{s}_i - \tfrac{1}{L_i} f'(\mathbf{s}_i)), \tag{6}$$

where $Ep_{\mathbf{Z}}(\cdot)$ represents the Euclidean projection onto the convex set $\mathbf{Z}$, which can be formulated as:

$$Ep_{\mathbf{Z}}(\mathbf{s}) = \arg\min_{\mathbf{x}\in\mathbf{Z}} \tfrac{1}{2} \| \mathbf{x} - \mathbf{s} \|_2^2. \tag{7}$$

So $\mathbf{x}_{i+1}$ is a gradient update of $\mathbf{s}_i$ and projected onto the constraint convex set $\mathbf{Z}$.

$\frac{1}{L_i}$ in Eq. (6) is the update stepsize at the i_{th} iteration and L_i is selected to satisfy the Armijo-Goldstein rule, i.e.,

$$f(Ep_{\mathbf{Z}}(\mathbf{s}_i - \tfrac{1}{L_i} f'(\mathbf{s}_i))) \leq f_{L_i,\mathbf{s}_i}(Ep_{\mathbf{Z}}(\mathbf{s}_i - \tfrac{1}{L_i} f'(\mathbf{s}_i))), \tag{8}$$

where $f_{L_i,\mathbf{s}_i}(\mathbf{x}) = f(\mathbf{s}_i) + \langle f'(\mathbf{s}_i), \mathbf{x} - \mathbf{s}_i \rangle + \frac{L_i}{2} \| \mathbf{x} - \mathbf{s}_i \|_2^2$ [10].

According to the above description, the accelerated algorithm based on Nesterov's method for our model is summarized in Algorithm 1:

Algorithm 1 An Accelerated Algorithm for Our Model.

Input: $\mathbf{A}_\tau$, L_0, $\mathbf{w}_0$, $\eta > 1$.
Output: $\mathbf{w}$.
Initialize $\mathbf{w}_1 = \mathbf{w}_0$, $t_{-1} = 0$, $t_0 = 1$,$i = 0$
while the convergence condition is not satisfied **do**
  $i = i + 1$, $t_i = \frac{1+\sqrt{1+4t_{i-1}^2}}{2}$, $\alpha_i = \frac{t_{i-2}-1}{t_{i-1}}$, $L = L_{i-1}$
  $\mathbf{s}_i = \mathbf{w}_i + \alpha_i(\mathbf{w}_i - \mathbf{w}_{i-1})$
  $\mathbf{w}_{i+1} = Ep_{\mathbf{Z}}(\mathbf{s}_i - \frac{1}{L}\mathbf{A}_\tau \mathbf{s}_i)$
  while $f(\mathbf{w}_{i+1}) > f_{L,\mathbf{s}_i}(\mathbf{w}_{i+1})$ **do**
    $L = \eta L$
    $\mathbf{w}_{i+1} = Ep_{\mathbf{Z}}(\mathbf{s}_i - \frac{1}{L}\mathbf{A}_\tau \mathbf{s}_i)$
  end while
  $L_i = L$
end while
$\mathbf{w} = \mathbf{w}_i$

In Algorithm 1, $f(\mathbf{w}) = \mathbf{w}^T \mathbf{A}_\tau \mathbf{w}$ and $f_{L,\mathbf{s}_i}(\mathbf{w}) = \mathbf{s}_i^T \mathbf{A}_\tau \mathbf{s}_i + (\mathbf{w} - \mathbf{s}_i)^T \mathbf{A}_\tau \mathbf{s}_i + \frac{L}{2} \| \mathbf{w} - \mathbf{s}_i \|_2^2$. Note that although there are various rules to update α_i in each iteration, here we follow the rule used by Liu *et al.* in [10].

3.3 Euclidean Projection

The remaining question is how to compute the Euclidean projection, i.e. $Ep_{\mathbf{Z}}(\mathbf{v})$, efficiently. In our model, the closed convex set $\mathbf{Z}$ is the probabilistic simplex, and $Ep_{\mathbf{Z}}(\mathbf{v})$ can be formulated as follows:

$$\begin{aligned} Ep_{\mathbf{Z}}(\mathbf{v}) = \quad & \arg\min_{\mathbf{w}} \tfrac{1}{2} \| \mathbf{w} - \mathbf{v} \|_2^2, \\ s.t.: \quad & \textstyle\sum_{i=1}^{D} w_i = 1, \\ & w_i \geq 0 \ \ (i = 1, ..., D). \end{aligned} \tag{9}$$

In order to describe the projection method more clearly, next, we give a simple derivation following the work in [5]. By means of the Lagrange multiplier, the above optimization problem can be reformulated as:

$$\mathcal{L}(\mathbf{w}, \lambda_0, \lambda) = \tfrac{1}{2} \| \mathbf{w} - \mathbf{v} \|_2^2 + \lambda_0 (\textstyle\sum_{i=1}^{D} w_i - 1) - \lambda^T \cdot \mathbf{w}, \tag{10}$$

where $\lambda_0 \in \mathbb{R}$ and $\lambda = [\lambda_1, \lambda_2, ..., \lambda_D]^T \in \mathbb{R}_+^D$ are Lagrange multipliers. Then:

$$\tfrac{\partial \mathcal{L}}{\partial w_i} = w_i - v_i + \lambda_0 - \lambda_i = 0, \tag{11}$$

we have:

$$w_i = v_i - \lambda_0 + \lambda_i. \tag{12}$$

According to KKT condition, $\lambda_i w_i = 0$. So if $w_i > 0$, then $\lambda_i = 0$, and $w_i = v_i - \lambda_0 + \lambda_i = v_i - \lambda_0$. It has been proven in [16] that if $v_i > v_j$ and $w_i = 0$, then $w_j = 0$. According to this, we denote $w_{(i)}$ as the i_{th} biggest element in $\mathbf{w}$ and suppose that there are d positive elements in total, so we have:

$$\textstyle\sum_{i=1}^{D} w_i = \sum_{i=1}^{d} w_{(i)} = \sum_{i=1}^{d} (v_{(i)} - \lambda_0) = 1, \tag{13}$$

where $v_{(i)}$ is the i_{th} biggest element in $\mathbf{v}$. Therefore,

$$\lambda_0 = \tfrac{1}{d} (\textstyle\sum_{i=1}^{d} v_{(i)} - 1). \tag{14}$$

Denote $\mathbf{u}$ is $\mathbf{v}$ sorted in descending order, then Eq. (14) can be rewritten as:

$$\lambda_0 = \tfrac{1}{d} (\textstyle\sum_{i=1}^{d} u_i - 1). \tag{15}$$

Now the question becomes how to find d. According to [16], the solution for d can be expressed as:

$$d = \max\{j \in [1, 2, ..., D] \mid u_j - \tfrac{1}{j} (\textstyle\sum_{i=1}^{j} u_i - 1) > 0\}. \tag{16}$$

In [5], Duchi *et al.* proposed an elegant ℓ_1-ball Euclidean projection method to find d and calculate λ_0 simultaneously, which has an expected time complexity of $O(D)$. Here we use this method to do the Euclidean projection onto the probabilistic simplex. This algorithm is summarized in Algorithm 2 [5].

Algorithm 2 Euclidean Projection Algorithm.

Input: $\mathbf{v} = [v_1, v_2, ..., v_D]^T \in \mathbb{R}^D$.
Output: $\mathbf{w} = [w_1, w_2, ..., w_D]^T \in \mathbb{R}^D$.
if $\sum_{i=1}^{D} v_i = 1$ and $\forall v_i \geq 0, i = 1, ..., D$ **then**
  $\mathbf{w} = \mathbf{v}$
  return
else
  Initialize $U = [1, ..., D]$, $s = 0$, $d = 0$
  while $U \neq \emptyset$ **do**
    select $k \in U$ at random
    $G = \{j \in U | v_j \geq v_k\}$, $L = \{j \in U | v_j < v_k\}$
    $\Delta d = length(G)$
    $\Delta s = \sum_{j \in G} v_j$
    if $(s + \Delta s) - (d + \Delta d) v_k < 1$ **then**
      $s = s + \Delta s$
      $d = d + \Delta d$
      $U = L$
    else
      $U = G \backslash \{k\}$
    end if
  end while
  $\lambda_0 = (s - 1)/d$
  $w_i = \max\{v_i - \lambda_0, 0\}, i = 1, 2, ..., D$
end if

3.4 Convergence Rate and Time Complexity Analysis

3.4.1 Convergence Rate Analysis

Note that the optimization problem derived from our model is an instance of a constrained smooth convex optimization problem, and it has been proven in [12] that when using Nesterov's method to solve a constrained smooth convex

optimization problem such as in Eq. (3), then following inequality holds:

$$f(\mathbf{w}_k) - f(\mathbf{w}^*) \leq \frac{2\widehat{L}_g \|\mathbf{w}_0 - \mathbf{w}^*\|_2^2}{k^2}, \quad (17)$$

where $\mathbf{w}^*$ is the optimal solution, $\mathbf{w}_0$ is the initial solution and $\widehat{L}_g = \max\{2L_g, L_0\}$. L_g is the Lipschitz continuous gradient of $f(\mathbf{w})$ and L_0 is the initial estimation of L_g [10]. Thus, the convergence rate of Algorithm 1 is $O(\frac{1}{k^2})$. If we define the desired accuracy as ϵ , then the iteration step needed to reach this accuracy is $O(\frac{1}{\sqrt{\epsilon}})$.

3.4.2 Time Complexity Analysis

The expected time complexity of Algorithm 2 is $O(D)$, where D is the dimension of $\mathbf{w}$. In each iteration of Algorithm 1, the main computational cost lies in steps 6, 7, 8 and 10. In steps 7 and 10, Algorithm 2 is called upon to do the Euclidean projection, and the expected time complexity is $O(D)$. In step 6, search point $\mathbf{s}_i$ is updated, and the time complexity for this step is also $O(D)$. In step 8, we need to calculate $f(\mathbf{w})$ and $f_{L,\mathbf{s}}(\mathbf{w})$, and the time complexity for this is $O(D^2)$. According to the above convergence rate analysis, there are $O(\frac{1}{\sqrt{\epsilon}})$ iterations in total. In addition, it takes $O(D^2T)$ time to calculate $\mathbf{A}_\tau$, which is the input of Algorithm 1. So the total time complexity of our approach is $O(\frac{D^2}{\sqrt{\epsilon}} + D^2T)$.

4. CONNECTION TO CCA

In this section, we first present the general idea and derivation of CCA, and then briefly discuss the difference between our method and CCA.

CCA uses two projection vectors, $\mathbf{w}_x$ and $\mathbf{w}_y$, to project $\mathbf{X}$ and $\mathbf{Y}$ respectively. CCA tries to maximize the correlation between these two projected series. $\mathbf{w}_x$ and $\mathbf{w}_y$ are the solution to the following optimization problem:

$$\begin{aligned} \arg\max_{\mathbf{w}_x, \mathbf{w}_y} \quad & \mathbf{w}_x^T \mathbf{X}\mathbf{Y}^T \mathbf{w}_y, \\ s.t.: \quad & \mathbf{w}_x^T \mathbf{X}\mathbf{X}^T \mathbf{w}_x = 1, \\ & \mathbf{w}_y^T \mathbf{Y}\mathbf{Y}^T \mathbf{w}_y = 1. \end{aligned} \quad (18)$$

Since our algorithm first centralizes and normalizes $\mathbf{X}$ and $\mathbf{Y}$, the objective function becomes similar to Eq. (2) if we constrain $||\mathbf{w}|| = 1$. The only difference is that CCA uses two projection vectors to project different sources while our algorithm only uses one. With this difference, we can better align two sides of text streams. As we discussed, the data from two sides are both text, which is homogeneous data. Therefore, if we use different projection vectors, it is less efficient to find common topics. Moreover, two projection vectors will mix up different leading keywords or topics.

5. EXPERIMENTS AND CASE STUDIES

In this section, we first use two toy data sets to demonstrate the advantages of our approach. Then we use a synthetic data set to demonstrate the efficiency of the accelerated algorithm. After that, we conduct experiments on two real data sets collected from Twitter[1] and Weibo[2]. The first data set is used to show our method can effectively track common trends and simultaneously select the corresponding keywords. The second data set is used to illustrate the application of our method on social influence analysis, i.e., follower link prediction.

[1]http://www.twitter.com
[2]http://www.weibo.com

5.1 Comparison with CCA: Two Toy Data Examples

First, we designed a toy data set to indicate where CCA fails to return the correct correlation when we only want to know the positive correlation factors. We generated two sets of data, each having 5 features and 50 samples, denoted as $\mathbf{X}_0$ (no lead-lag) and $\mathbf{Y}$ respectively. Thus, we had $\mathbf{X}_0 \in \mathbb{R}^{5\times 50}$ and $\mathbf{Y} \in \mathbb{R}^{5\times 50}$. The first feature was positively correlated, and the last feature was negatively correlated, as shown in Figure 1(a). Other features in $\mathbf{X}_0$ and $\mathbf{Y}$ were randomly sampled from a Gaussian distribution and independent from each other. The projection vector generated by our method is shown in Figure 1(b) and the two projection vectors returned by CCA are shown in Figure 1(c). We can see that our algorithm successfully found that the first feature triggered the positive correlation and assigned a high weight to it. However, in this case, CCA assigned the highest values to the last feature, which in fact is negatively correlated. The result is quite intuitive since CCA does not consider whether the correlation is positive or negative while ours, with non-negative constraints, can effectively figure out only the positive correlation.

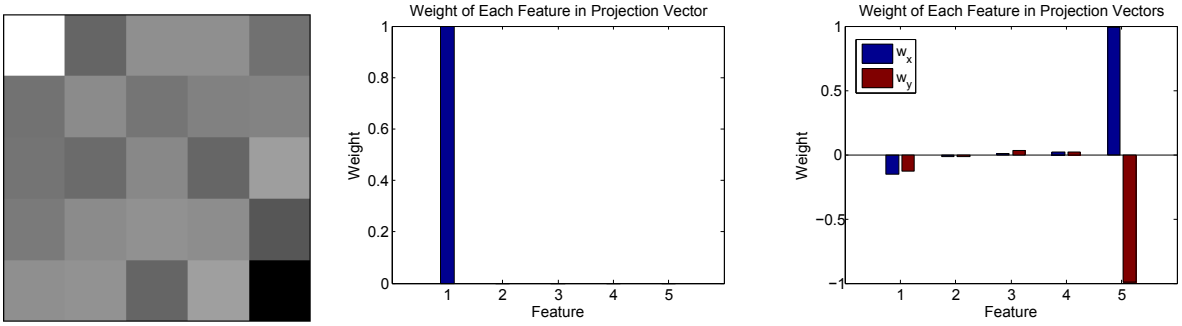
The second toy experiment is used to show that CCA may mix up different features while our method does not. There are three features in this data, which are denoted as f_1, f_2 and f_3. Only f_3 is weakly correlated. However, f_1 in $\mathbf{Y}$ is strongly correlated with f_2 in $\mathbf{X}$, as shown in Figure 2(a). Figure 2(b) shows that our method successfully found the common feature f_3, which was weakly correlated, while CCA gave higher weights to features f_1 in $\mathbf{w}_y$ and f_2 in $\mathbf{w}_x$ and concluded that the two data are strongly correlated, as illustrated in Figure 2(c).

The results of these two experiments imply that CCA can mix up positive and negative correlations as well as detect cross-topic correlations, which is undesirable in many text mining scenarios. In contrast to CCA, our method successfully detects the correct positive correlation in both experiments, thus showing the advantage of our method over CCA in these instances.

5.2 Correctness and Runtime of Accelerated Algorithm

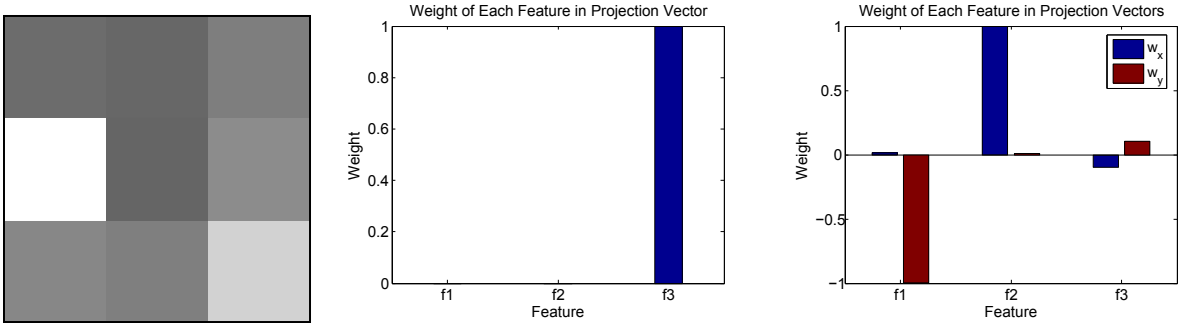
In order to validate the correctness and efficiency of the accelerated algorithm, we conducted experiments on a synthetic data set, where every element of $\mathbf{A}_\tau$ was set to an independent gaussian random number. The baseline method used here was the extended gradient method which was proposed in [7]. Both methods were implemented using MATLAB 2009b and all the experiments were performed on a computer running Windows 7 with Intel Core 2 Quad CPU(2.66 GHz) and 4G RAM. We ran both methods on synthetic data 10 times and report the average results in Figure 3.

Figure 3(a) shows the time consumed by our accelerated method and gradient method to reach the same accuracy at different dimensions. In this experiment, the length of time series was set to a fixed number (5000). From Figure 3(a), we can see that the time consumed by both methods grew in a way similar to a quadratic function as the dimension increases. However, the time consumed by our accelerated method grew more slowly. In addition, our accelerated



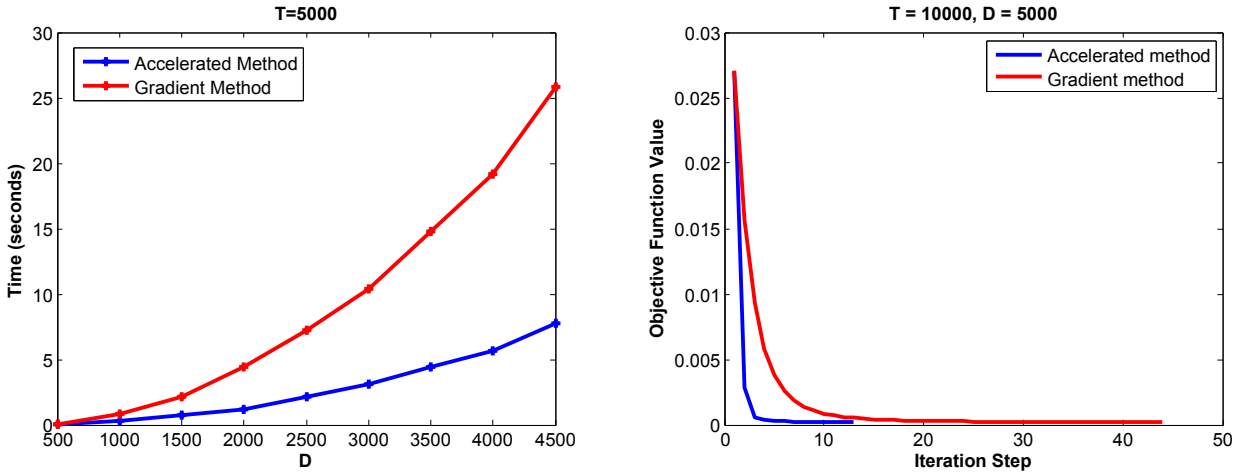
(a) Pairwise Correlation. (b) Our Results. (c) CCA Results.

Figure 1: Toy data example on positive and negative correlations. Figure 1(a) shows the pairwise correlation values between each feature of X and Y. Every row corresponds to a feature of X. From top to bottom the rows are the first feature to the last feature. Every column corresponds to a feature of Y. From left to right the columns are the first feature to the last feature. The pixel value is proportional to the correlation value, i.e. 255 corresponds to 1 and 0 corresponds to -1. So the darker the color is, the smaller the correlation value is. Figure 1(b) and Figure 1(c) are the results generated by our method and CCA respectively.



(a) Pairwise Correlation. (b) Our Results. (c) CCA Results.

Figure 2: Toy data example of mixture of cross correlation. Figure 2(a) shows the pairwise correlation value between each feature of X and Y, similar to Figure 1(a). Figure 2(b) and Figure 2(c) are the results generated by our method and CCA respectively.



(a) Time. (b) Objective Function Value.

Figure 3: Experimental results with the synthetic data. Figure 3(a) shows the time consumed by our accelerated method and gradient method to reach the same accuracy with different dimensions. Figure 3(b) shows the objective function value of our accelerated method and gradient method in different iteration steps.

method clearly took less time than the gradient method in the same dimension, which implies the advantage of the accelerated method in large-scale applications. Figure 3(b) shows the objective function values of both methods at each iteration, with 5000 dimensions and 10,000 time points. It is clear that the objective function value of accelerated method decreased much faster than that of the gradient method and needed less iteration steps to achieve the same accuracy, thus once again validating the advantage of our accelerated method over the gradient method.

5.3 Twitter: Common Trend Detection and Keyword Selection

To validate our method on real-world data, we collected tweets from Twitter that were related to the 2012 US presidential election and got 71,446,627 tweets from 9,667,382 users. A standard stopword list was used to remove stopwords. After stemming, we extracted the bag-of-words features for each tweet and assigned every keyword their TFIDF values. In this experiment, we used our method to detect the common topic trends shared by two different Twitter users and selected the keywords to identify the common topic simultaneously.

For example, "LOLGOP" is a popular Twitter user who cares about politics and "POLITICO" is a professional Twitter account which tracks and reports political events. We extracted the bag-of-words features of tweets posted by LOLGOP and POLITICO during October of 2012 and split them by hour, then we got two multidimensional sequential datasets, denoted as $\mathbf{X}$(LOLGOP) and $\mathbf{Y}$(POLITICO) respectively. In order to eliminate the noise caused by keywords that appear very infrequently, we filtered keywords that appear less than one percent of the total time points, based on the intuition that very rare keywords are not discussed a lot and have less possibility to trigger trends. The common topical trend returned by our method is shown in Figure 4. It illustrates that they match with each other very well at several time points. For example, we can see the three highest peaks correspond to three TV debates between Obama and Romney on Oct 3rd, Oct 16th and Oct 22nd. Moreover, the words that the projection vector assigns the highest values to are {*think: 0.051, cbs: 0.043, vote: 0.041, love: 0.037 , china: 0.035, debate: 0.034, watch: 0.034* }. These words do have some connections with the presidential election, and can represent common topics.

5.4 Twitter: Trend Setter Detection

In this experiment, we used the same dataset as the previous experiment and applied our method to detect trend setters, that is, those whose tweet content can predict what will be discussed later on Twitter. The 2000 most retweeted users and all their tweets posted from Oct 1st to Oct 31th in 2012 were selected. For each user, the bag-of-words features of his/her tweets were extracted and split by hour, denoted as $\mathbf{X}_\tau$ and those of all the others' tweets as $\mathbf{Y}$. Similar to the previous experiment, stopwords and words appearing very infrequently were removed.

In order to discover how well a certain user can predict the tweet content of the public afterwards, and how far ahead of time he/she can predict, we traversed τ from 0 to 5, indicating that our search range was within five hours. For each pair of users' tweets, we used a five-fold cross validation method to find the best τ. The data was split into training

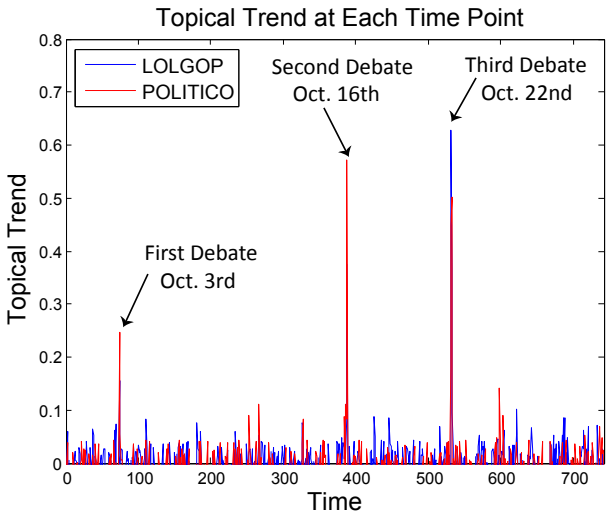


Figure 4: Common topical trends of users "LOLGOP" and "POLITICO".

sets $\{\mathbf{X}_{\tau,train}$ and $\mathbf{Y}_{train}\}$ and validation sets $\{\mathbf{X}_{\tau,validate}$ and $\mathbf{Y}_{validate}\}$. We applied our method to the training data sets to obtain the projection vector. Then we used this projection vector to map $\mathbf{X}_{\tau,validate}$ and $\mathbf{Y}_{validate}$ into two one-dimensional series $\mathbf{x}$ and $\mathbf{y}$. We calculated the average correlation score of $\mathbf{x}$ and $\mathbf{y}$ to select the best τ, denoted as τ_{best}. A higher correlation and larger lead-time mean that the user is more influential. It is more likely that this user is the source of some important information or an opinion leader. Thus we defined a new criteria, the Prediction Ability (PA), to measure the predictability of a user. For a certain user, we selected the lead-time τ_{best} corresponding to the highest correlation and calculated the PA score according to $\tau_{best} \times correlation_{\tau_{best}}$. Then we ranked all users according to their PA scores. Table 1 shows the top users ranked by PA scores from our data set.

Rank	User Name	τ_{best}	Corr	PA
1	Wall Street Journal	4	0.2028	0.8112
2	ABC News	2	0.3589	0.7178
3	The Caucus	2	0.3543	0.7086
4	jimgeraghty	2	0.3018	0.6036
5	Yahoo news	2	0.2819	0.5638
6	Jason Easley	1	0.5507	0.5507
7	Truth Tweeter	1	0.5357	0.5357
8	Greta Van Susteren	1	0.4620	0.4620
9	The Associated Press	1	0.4601	0.4601
10	Chuck Woolery	1	0.4230	0.4230

Table 1: Trend setters and their ranking scores.

From Table 1, we can see that the Twitter user with the highest PA score was "Wall Street Journal," a popular Twitter account that posts frequently the latest news stories. Moreover, "ABC News," "The Caucus," owned by the NY Times, and "Yahoo! news" are already well-known for public news. "The Associated Press" is also a news account that posts frequently on the latest news . Thus, the most influential trend setters are news media accounts. This may be

User	Profile
jimgeraghty	"NR Campaign correspondent."
Jason Easley	"Proud liberal website" and "In a world full of corporate cash, we are independent."
Truth Tweeter	"I love America. I love our Constitution. I hope we return to our LIBERTARIAN values!"
Greta Van Susteren	"Host of ON THE RECORD - Weeknights at 10p & 1a ET on Fox News Channel."
Chuck Woolery	"Political Activist."

Table 2: Users' profiles that are not popular news providers in Table 1.

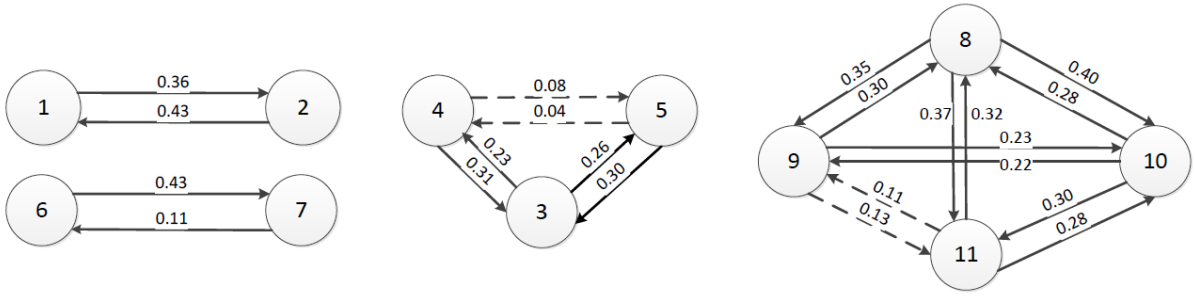


Figure 5: Examples of strongly influential relationships.

because these news outlets are closer to the source of political news or more sensitive to this news. Moreover, the other users' profiles that are not news providers are shown in Table 2. We checked their tweets and found most of their topics were related to political events. The trend setter detection results validate that our algorithm can return meaningful trend setters and is useful for social influence analysis.

From Table 1, we can see that the Twitter user with the highest PA score was "Wall Street Journal," a popular Twitter account that posts frequently the latest news stories. Moreover, "ABC News," "The Caucus," owned by the NY Times, and "Yahoo! news" are already well-known for public news. "The Associated Press" is also a news account that posts frequently on the latest news. Thus, the most influential trend setters are news media accounts. This may be because these news outlets are closer to the source of political news or more sensitive to this news. Moreover, the other users' profiles that are not news providers but with higher PA scores are shown in Table 2. We checked their tweets and found most of their topics were related to political events. The trend setter detection results validate that our algorithm can return meaningful trend setters and is useful for social influence analysis.

5.5 Weibo: Influence Relationship Analysis

To perform this experiment, we crawled a data set that contains both short messages and user graph information from Weibo.com, a famous social network website in China which is similar to Twitter. We used the user links (i.e., follower and followee links) to verify the correlation analysis results. There were a total of 56,290,313 messages from 1,599,795 users. For each Weibo message, the stopwords were removed according to a standard Chinese stopword list and the bag-of-words features were extrated. Assume that there were two users, A and B, and we wanted to find whether A follows B. We set the temporal lead value of B as one day, which means user B's messages were used to predict A's messages the next day. If B can predict A's messages, then we predict that B influences A, or A is a follower of B.

Some of the example influence relationships found by our method are shown in Figure 5. We anonymized the users' names here since they are all in Chinese and represent them with numbers. We found the following:

- User 1 was a shopping website, and user 2 was the founder of this website. We checked these two users and found that they follow each other and always retweet each other's messages. Our algorithm identified that they are highly correlated.

- Users 4 and 5 were two accounts that frequently publish material about oral English study, and user 3 was one of their followers. Since user 3 often refered to content similar to that found in the messages of users 4 and 5, users 4 and 5 have a strong influence on user 3. It is therefore reasonable to conclude that user 3 is very interested in learning oral English, and she/he would follow more users who are talking about learning oral English.

- User 6 was a university in Beijing and user 7 was a class held in this university, and they follow each other. However, user 6 could predict user 7's content quite well while user 7 is relatively more difficult to predict user 6's. This indicates that user 6 has a stronger influence on user 7. By checking the content we found that user 6 frequently posts some news related to this university and user 7 cares about these news very much. The results returned by our method reflect the real influence between these two users.

- User 8 was a very popular Weibo account that frequently posts about the latest news, thoughts about life, or common sense about health care. These messages were widely re-tweeted by its followers. Users 9, 10 and 11 are three of his/her followers, who are strongly influenced by user 8. Moreover, users 9/10 and users 10/11 are mutual followers. From Fig. 5, we can see that the weights were higher on the links

where follower relationships existed, and the weights were lower where follower relationships were absent.

All the above examples show that follower links are useful for verifying our correlation results. Beyond that, our method can be further used to predict the possible followers of a user and find the true influence relationship.

To compare our method with CCA, we selected the top 1,162 users ranked by the numbers of their followers and extracted all their messages. We split the messages by day and built text streams for each user. There were 3,355 follower links in total between these users. We also picked 3,355 randomly selected user pairs where the follower relationships did not exist. Then we used the correlation results of our method and CCA to predict the links between users. The prediction was evaluated by ROC curve. We plotted the true positive rate against the false positive rate and got the ROC curves, which are shown in Figure 6. Moreover, the AUC (area under the ROC curve) of our method was 0.7657, higher than that of CCA, which was 0.6236. We can see that our algorithm can significantly better predict the follower links in this data.

In addition, we observed in the experiment that in many cases where CCA assigns high correlation scores, the two users have no link between each other and they tend to talk about obviously different topics. The two projection vectors returned by CCA also differ with each other significantly in these cases. Thus, although the text streams from these two users are not relevant, CCA still detects that they are highly correlated, which is spurious. The high correlation scores may be caused by similar posting patterns, for example, when both users post a large number of messages suddenly at adjacent time points. In the results returned by our methods, this kind of case was very rare. This experiment further demonstrates that CCA can find false correlations between text streams, while our method can effectively reduce such correlations, since we use only one projection vector and impose sparse non-negative constraints on it.

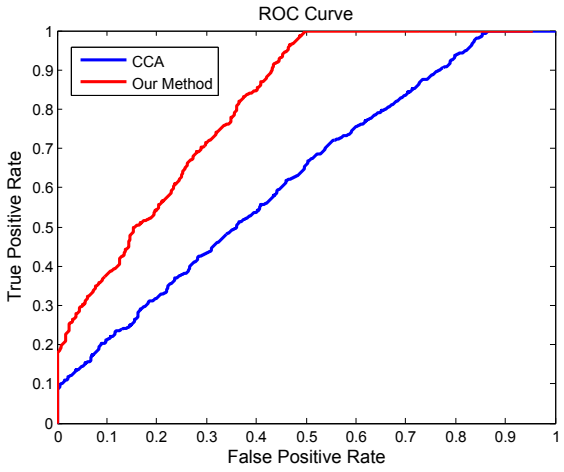


Figure 6: ROC curves of our method and CCA for prediction of follower links.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an approach to analyzing the lead-lag correlation behavior between two text streams. In addition to finding the correlation, we also discovered the keywords that trigger the correlation. To this end, we formulated the problem as a least square regression problem over the projected time series of two text streams. To find the correlated keywords, we imposed non-negative and sparse constraints over the vector elements. To solve this optimization problem, we reported an accelerated gradient algorithm based on Nesterov's method, which is able to find the optimal projection vector. We conducted several experiments and case studies on both synthetic data and real data to demonstrate the advantage and capabilities of our approach.

Although our approach has the ability to overcome some drawbacks of CCA and is useful in finding common trends, detecting trend setters and predicting follower links, it still has some limitations. First, our approach treats different keywords equally. However, different keywords may have different abilities to trigger topic trends, so it would be more beneficial to assign different weights to different keywords. This can be solved by introducing a Bayesian model of regression in which weights can be regularized with hyper-parameters. When there are multiple pairs of text streams, the hyper-parameters can be estimated to adjust the corpus. Second, currently we set the lead-lag time as a constant and find a global lead-lag between two text streams. We are also interested in finding local lead-lag patterns at individual time points. We will extend our model to automatically optimize the time-variant lead-lag parameters to align two text streams.

7. ACKNOWLEDGEMENTS

The authors would like to thank Tianyi Wang for helping us crawl the Weibo data. This research work is supported by 863 Program of China under Grant No. 2012AA011004 and Initiative Scientific Research Program of Tsinghua University under Grant No. 20111081023.

8. REFERENCES

[1] J. Allan. Topic detection and tracking. chapter Introduction to topic detection and tracking, pages 1–16. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[2] F. Bießmann, J.-M. Papaioannou, M. Braun, and A. Harth. Canonical trends: Detecting trend setters in web data. In *ICML*, 2012.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[4] X. Chen, W. Pan, J. T. Kwok, and J. G. Carbonell. Accelerated gradient method for multi-task sparse learning problem. In *ICDM*, pages 746–751, 2009.

[5] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *ICML*, pages 272–279, 2008.

[6] R. F. Engle and C. W. J. Granger. Co-integration and error correction: Representation, estimation and testing. *Econometrica*, 55(2):251–276, 1987.

[7] S. Ji and J. Ye. An accelerated gradient method for trace norm minimization. In *ICML*, pages 457–464, 2009.

[8] K. Lê Cao, P. Martin, C. Robert-Granié, and P. Besse. Sparse canonical methods for biological data integration: application to a cross-platform study. *BMC bioinformatics*, 10(1):34, 2009.

[9] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD*, pages 497–506, 2009.

[10] J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient l2, 1-norm minimization. In *UAI*, pages 339–348, 2009.

[11] J. Liu and J. Ye. Efficient euclidean projections in linear time. In *ICML*, pages 657–664, 2009.

[12] A. Nemirovski. Efficient methods in convex programming. 1994.

[13] Y. Nesterov. Gradient methods for minimizing composite objective function, 2007.

[14] E. Parkhomenko, D. Tritchler, and J. Beyene. Genome-wide sparse canonical correlation of gene expression with genotypes. In *BMC proceedings*, volume 1, page S119, 2007.

[15] E. Parkhomenko, D. Tritchler, and J. Beyene. Sparse canonical correlation analysis with application to genomic data integration. *Statistical Applications in Genetics and Molecular Biology*, 8(1):1–34, 2009.

[16] S. Shalev-Shwartz and Y. Singer. Efficient learning of label ranking by soft projections onto polyhedra. *J. Mach. Learn. Res.*, 7:1567–1599, Dec. 2006.

[17] X. Shi, R. Nallapati, J. Leskovec, D. A. McFarland, and D. Jurafsky. Who leads whom: Topical lead-lag analysis across corpora. In *NIPS Workshop*, 2010.

[18] Y. Song, S. Pan, S. Liu, F. Wei, M. X. Zhou, and W. Qian. Constrained coclustering for textual documents. In *AAAI*, 2010.

[19] G. Steeg and A. Galstyan. Inferring predictive links in social media using content transfer. In *WSDM*, 2013.

[20] J. Tang, J. Sun, C. Wang, and Z. Yang. Social influence analysis in large-scale networks. In *KDD*, pages 807–816, 2009.

[21] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[22] X. Wang, C. Zhai, X. Hu, and R. Sproat. Mining correlated bursty topic patterns from coordinated text streams. In *KDD*, pages 784–793, 2007.

[23] X. Wang, K. Zhang, X. Jin, and D. Shen. Mining common topics from multiple asynchronous text streams. In *WSDM*, pages 192–201, 2009.

[24] D. M. Witten, T. Hastie, and R. Tibshirani. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 2009.

[25] D. M. Witten and R. J. Tibshirani. Extensions of sparse canonical correlation analysis with applications to genomic data. *Statistical Applications in Genetics and Molecular Biology*, 8(1):28, 2009.

[26] J. Zhang, Y. Song, C. Zhang, and S. Liu. Evolutionary hierarchical Dirichlet processes for multiple correlated time-varying corpora. In *KDD*, pages 1079–1088, 2010.